Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matic Trebušak

Razvoj mobline aplikacije za vzpodbujanje dobrih del

diplomsko delo

na univerzitenem študijskem programu
računalništva in informatike

Mentor: doc. dr. Tomaž Hovelja

Ljubljana, 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Kandidat naj razvije mobilno aplikacijo za vzpodbujanje dobrih del. Kandidat naj za uspešen razvoj preuči potrebno literaturo o metodologijah razvoja programske opreme in primernih tehnologijah za razvoj mobilnih aplikacij. Na osnovi tako pridobljenega znanja naj razvije potrebne rešitve, ki bodo aplikaciji omogočale izvajanje ključnih funkcionalnosti. Kandidat naj pri tem poizkuša aplikacijo razviti na način, ki bo omogočal in vzpodbujal oblikovanje socialne mreže uporabnikov aplikacije.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Matic Trebušak, z vpisno številko **63080133**, sem avtor diplomskega dela z naslovom:

Razvoj mobline aplikacije za vzpodbujanje dobrih del

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Tomaža Hovelje,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 2. oktobra 2014 Podpis avtorja:

Na prvem mestu bi se rad zahvalil staršem, ki so me s svojo ljubeznijo vodili skozi življenje, da sem lahko zaključil študij, širši družini, ki mi je vedno nudila toplo in produktivno okolje, ter vsem prijateljem, ki so mi vsak na svoj način lepšali študijske dni.

Posebna zahvala gre tudi mentorju doc. dr. Tomažu Hovelji za usmerjanje in uporabne nasvete pri pisanju diplomskega dela.

Ljudem, ki so pozabili biti ljudje.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
XML	Extensible Markup Language	Razširljiv označevalni jezik
JSON	JavaScript Object Notation	Objektna notacija JavaScript
PHP	PHP Hypertext Preprocessor	Tričrkovni rekurzivni algoritem
HTML	HyperText Markup Language	Jezik za označevanje nadbesedila
SQL	Structured Query Language	Strukturirani povpraševalni jezik
ADT	Android Development Tools	Razvojna orodja za Android

Povzetek

V sklopu diplomske naloge je bila razvita mobilna aplikacija za vzpodbujanje dobrih del. Zasnovan je bil koncept albuma v katerem dejanja predstavljalo strani v albumu. Uporabniku je omogočeno opravljanje dejanj, dodajanje slik, deljenje dejan, pregled statistike in druge možnosti. Pri opravljanju je med drugim poudarek tudi na mreženju uporabnikov. Tako se lahko povežejo s prijatelji in dejanja opravljajo skupaj. Predlagajo lahko tudi nova dejanja, ki so na voljo ostalim. Aplikacija shranjuje podatke lokalno, prav tako se poveže s spletom in jih shrani na strežnik. Uporabljene so podporne knjižnice, ki omogočajo uporabo funkcionalnosti na starejših različicah operacijskega sistema Android. Podprto je tudi komuniciranje z ostalimi aplikacijami na mobilni napravi.

Ključne besede: Android, mobilne aplikacije, dobra dela, vzpostavljanje socialnih omrežij

Abstract

In the thesis we developed mobile application for the promotion of good deeds. It was conceived as album in which deeds present a page in the album. The user is able to perform the deeds, add pictures, share deeds, examine the statistics and other options. There is an additional emphasis on networking between users. They can connect with friends and preform actions together. They can also propose new deeds that are available to others. The application stores data locally, but can also connect to the internet and store them on the server. The supporting libraries are included so that users can use the functionalities in older versions of the Android operating system. Option for communicating with other applications on the mobile device is included as well.

Keywords: Android, mobile application, good deeds, social networking

Poglavje 1

Uvod

Ob koncu študija na Fakulteti za računalništvo in informatiko sem se spraševal, na katerem računalniškem področju se vidim v prihodnosti. Teh je res ogromno. Vedel sem, da bi rad posvetil svoje diplomsko delo temi, ki bi mi služila kot korak naproti izbranemu področju. Po končanih izpitih sem se za nekaj mesecev odpravil na popotovanje, da bolje spoznam sebe in razmislim o svoji prihodnosti, saj se mi je to zdelo izjemo pomembno. Potoval sem in razmišljal, nekje med trekingom po Nepalu, se mi je posvetilo. Ustvarjati želim mobilne aplikacije. Vedel sem, da o njih ne vem veliko, a ideja o morju možnosti, ki se odpirajo z revolucijo pametnih telefonov, me je prevzela. Širina problemov, ki jih lahko naslovimo z mobilnimi aplikacijami, mi ponuja prav to dinamičnost, ki sem jo iskal. Telefone in tablice imamo dandanes s seboj praktično povsod. A zavedal sem se tudi omejitev. Konkurenca na tem področju je velika. Tudi bliskovit razvoj tehnologije mi daje vedeti, da to ni služba za večno in da bo potrebno velikokrat popraviti smer, kamor potujem. A navsezadnje, katero delo v računalništvu ni takšno?

V današnjem času smo dnevno bombardirani z informacijami na vsakem koraku. Kot posledica tega se je v nas razvil obrambni mehanizem, ki nam omogoča prezreti kopico idej in izdelkov, s katero smo zasuti. Med zavednim in nezavednim ločevanjem zeli od plevela nas ta ignoranca včasih zapelje na

pot, nekoliko odmaknjeno od naših vrednot. Vsakodnevni vrvež in moderno hitenje od enega opravila do drugega nas počasi, odnaša v reko egoizma, kjer imamo vse manj volje in časa za pomoč sočloveku. Kot socialna bitja nas ta reka počasi a vztrajno pelje v osamljenost, njen tok pa postaja vse močnejši. Namen dela je razviti aplikacijo, ki bo uporabnike mobilnih aplikacij vzpodbudila k opravljanju dobrih del.

Za realizacijo namena sem si zadal cilj ustvariti zanimivo aplikacijo, ki bi uporabniku vse to omogočala. Tako sem se domislil koncepta albuma, kjer bi uporabnik dobil ideje in v katerega bi lahko shranjeval svoja dobra dejanja. Premikanje po straneh albuma mora biti gladko in preprosto, smiselno pa je uporabiti fragmente. Ob zagonu aplikacije se morajo podatki ohraniti, zato bo potrebno tudi shranjevanje na napravo. Privlačnost aplikacije se poveča, če lahko uporabniki dejanja opravljajo skupaj, zato je potrebno zgraditi sistem, kjer bo aplikacija komunicirala s spletom. Za shranjevanje podatkov na spletu bo potrebno ustvariti bazo vnosov, iz katere in v katero bo aplikacija lahko pisala. V koncept albuma je primerno vgraditi tudi sposobnost zajemanja in izbiranja uporabniških fotografij. Za prepoznavnost aplikacije in večanje bazena uporabnikov bi bilo potrebno ustvariti tudi način za deljenje dobrih dejanj in ozaveščanje ostalih. Možnosti in predlogov za dejanja je zares ogromno, zato je smiselno omogočiti uporabnikom, da lahko tudi sami prispevajo svoja dejanja. Na koncu je dobro imeti tudi pregled in možnost upravljanja predlaganih dejanj kot administrator. S takšnim konceptom v mislih sem se lotil razvoja.

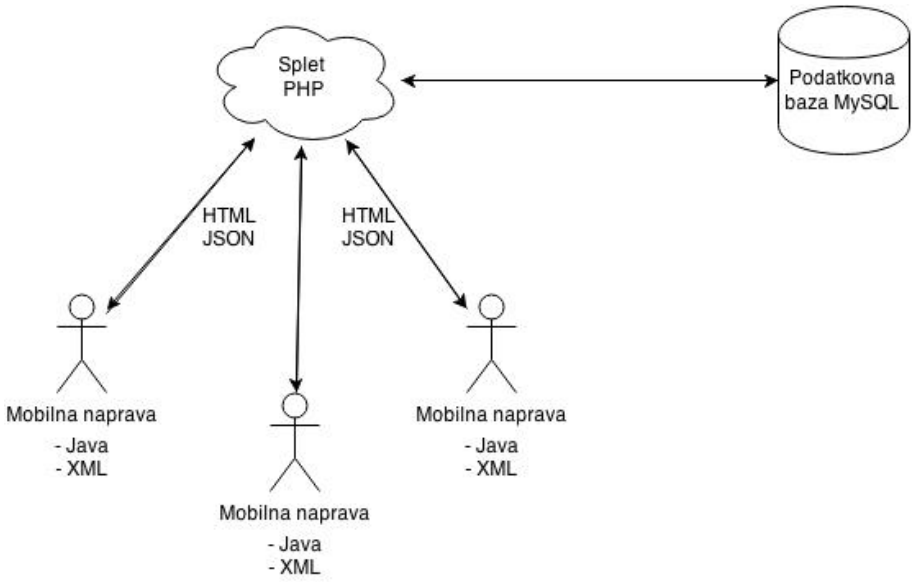
Poglavje 2

Uporabljene tehnologije, metodologija in razvojno okolje

Za izpolnitev zastavljenih ciljev sem se morali poslužiti različnih tehnologij, opisanih v nadaljevanju. Aplikacijo za mobilno napravo sem napisal v programskem jeziku Java, izgled aplikacije pa je definiran z datotekami tipa XML. Celotni razvoj je potekal v okolju Eclipse, prilagojenem za razvoj Androida. Naprave komunicirajo s spletom preko protokola HTML, povratne informacije pa so poslane kot objekti JSON. Na spletu so naložene skripte PHP, ki poslane informacije sprejmejo, nato pa izvedejo ustrezne poizvedbe v bazi v jeziku SQL. Baza vrne odgovor in stran ga ustrezno zapakira za prenos na napravo. Dostop do baze je možen tudi preko orodja phpMyAdmin, ki nam olajša urejanje podatkov preko grafičnega vmesnika. Simbolni prikaz je na sliki 2.1.

2.1 Metodologija

Pri izdelovanju aplikacije sem se opiral na osnovna načela agilnih metodologij [4]. Pri razvoju je bil uporabljen iterativni pristop. Delujočo programsko opremo sem uvrstil pred popolno dokumentacijo. Komunikacija med člani ekipe je bila poenostavljena kolikor se da, saj sem projekt razvijal sam. Kot



Slika 2.1: Prikaz povezave naprav z bazo.

pogovor z naročnikom sem smatral dogovarjanja glede izboljšav z mentorjem ter prijatelji, katere sem prosil naj mi opišejo svojo uporabniško izkušnjo s prototipom razvite aplikacije. Tako sem lahko upošteval njihove nasvete in se prilagodil spremembam. Vsaka iteracija je potekala tako, da sem si izbral primer uporabe, ki ga želim realizirati. Preučil sem več možnosti, kako priti do rešitve, in izbral tisto, ki se mi je zdela najustreznejša. Sledilo je testiranje na napravah in iz rezultatov sem lahko naredil ustrezne popravke. Enostavnost procesa je vodila kot ključ do uspeha. Zgodba se je skozi iteracije sestavljala, dokler ni prototip dobil končne podobe. Sledil bi lahko test na večji množici uporabnikov in krpanje morebitnih lukenj v aplikaciji. Iz literature sem prav tako pridobil koristne informacije o lastnostih socialnih omrežjih [6, 2]. V aplikaciji sem tako zajel elemente, ki vzpodbujajo povezovanje ljudi in jo naredijo uporabnikom zanimivejšo. Upošteval sem priporočila glede personalizacije albuma, popestritve aktivnosti s slikami, sodelovanja uporabnikov, možnost opazovanja napredka ostalih uporabnikov, vključenosti uporabnikov v skupnost ter širjenje svojih opravljenih dejanj med prijatelje.

2.2 Razvojno okolje Eclipse in ADT

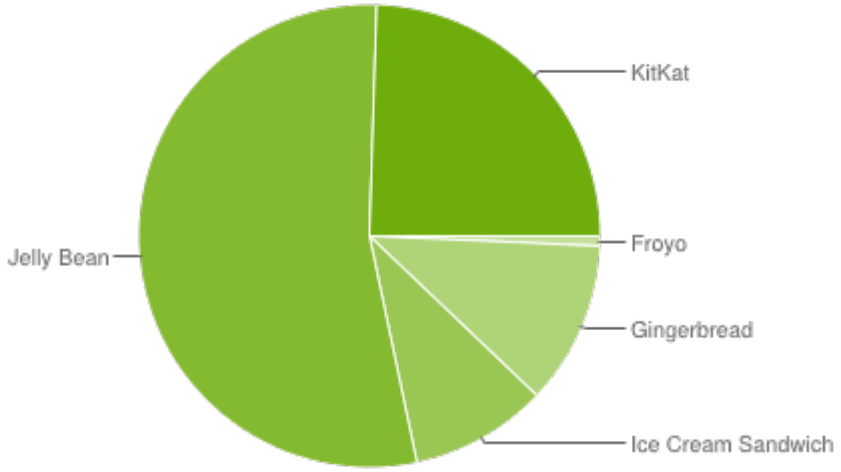
Razvojno okolje, v katerem sem razvil aplikacijo, je Eclipse, različica 8.0, v katerem so integrirana orodja za razvoj aplikacij Android pod skupno kratico ADT. Uporabljal sem vtičnik ADT, različico 22.6.1. Uveljavljena praksa je, da programer namesti razvojni paket, v katerem je vse, kar potrebuje za razvoj aplikacij za operacijski sistem Android, nato pa namesti dodatke za razvoj, v kolikor jih potrebuje. V prihodnosti se načrtuje prehod na razvojno okolje narejeno posebej za aplikacije Android, imenovano Android Studio. Zanj nisem odločil, ker je še v fazi testiranja [16, 18].

2.3 Programski jezik Java

Glavni jezik, v katerem se pišejo aplikacije za delovanje na operacijskem sistemu Android, je Java. Razvojno okolje olajša programiranje z integrirano dokumentacijo, avtomatičnim dopolnjevanjem ukazov, preverjanjem črkovanja ... Za razvoj je zaenkrat primernejša različica Java JDK 6.0, saj je bolje podprta.

2.4 Operacijski sistem Android

Android je najpopularnejši operacijski sistem za mobilne naprave. Nameščen je na več sto milijonov naprav po celem svetu in vsak dan se aktivira več kot milijon naprav s tem operacijskim sistemom. To je glavni razlog za izbor te platforme. V kratkem času je doživela razcvet in tako kot ena prvih na trgu postala vodilna. Googlova ekipa je razvila močan sistem, ki temelji na Linuxovem jedru. Gre za odprtokodni operacijski sistem, tako da je koda na vpogled vsem [18]. Pri razvoju aplikacije sem zajel kompatibilnost z različico Androida vse do 2.2, kar v seštevku z vsemi vmesnimi različicami do zadnje 4.4 znese skoraj 100 % vseh naprav, ki uporabljajo Android [10], kot je prikazano na sliki 2.2. Pri tem sem si pomagal s podpornima knjižnicama v4 in v7.



Slika 2.2: Podatki so zbrani v sedemdnevnem obdobju, ki se je končalo dne 9. 9. 2014. Vse verzije z manj kot 0.1 % deležem niso prikazane.

2.5 Označevalni jezik XML

Za definiranje izgleda aplikacije in postavitve elementov v njej so uporabljene ločene datoteke tipa XML. Za njih je značilna hierarhična ureditev. Razvojno okolje, ki ga pretočimo s spleta, že vsebuje vtičnike za obdelavo datotek XML.

2.6 Objektna notacija JSON

JSON je odprtokodni format za prenos podatkovnih objektov med uporabnikom in strežnikom v paru vrednost-ključ. Služi kot alternativa formatu XML in je uporabljen v izdelani aplikaciji [24].

2.7 Sistem za upravljanje s podatkovnimi bazami MySQL

Podatki so na strežniku shranjeni v bazi. Za implementacijo baze je bila uporabljena odprtokodna relacijska podatkovna baza MySQL, ki uporablja

jezik SQL [23]. Gre za MySQL različico 5.1.

2.8 Programski jezik PHP

PHP je skriptni programski jezik, ki je pri razvoju aplikacije služil kot vmesni člen med komunikacijo z mobilno napravo in bazo podatkov na strežniku [22]. Uporabljena je bila različica 5.4.28.

2.9 Orodje phpMyAdmin

Orodje phpMyAdmin je namenjeno za upravljanje baze. Gre za brezplačano programsko opremo, ki poenostavi urejanje baze s svojim grafičnim vmesnikom [21]. Za administracijo baze sem pri razvoju obravnavane aplikacije uporabil orodje phpMyAdmin, različico 3.4.3.1, ki je na voljo pri ponudniku gostovanja Agilityhoster, kjer je sama baza tudi postavljena.

Poglavje 3

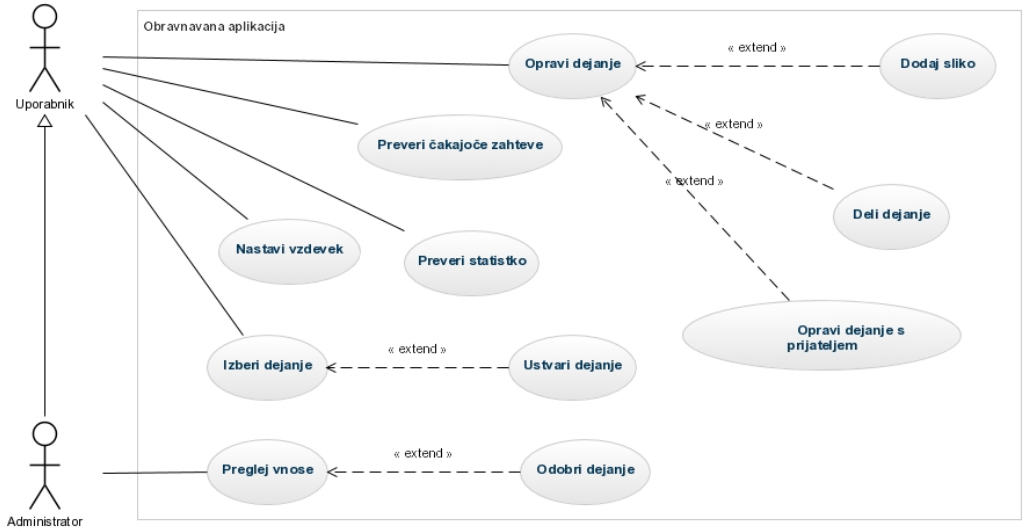
Aplikacija

3.1 Delovanje aplikacije

Koncept aplikacije je zasnovan na albumu dobrih del. Vsakemu dobremu delu pripada svoja stran. Vsak uporabnik najprej podpiše album s svojim vzdevkom, s čimer mu doda osebno noto, ter v podzavesti začne ustvarjati navezanost. Nato lahko začne polniti strani albuma.

Na začetku so mu na voljo dejanja, ki jih lahko konča sam. Ko izmed ponujene množice najde nekaj, kar bi rad opravil, in se tega loti, se zraven zapiše datum začetka. Ob končanem dejanju ga uporabnik potrdi. Zraven lahko doda fotografijo, ki jo je posnel ob izvajanju. Stran v albumu je tako zapolnjena in uporabniku ostane spomin na dobro delo, ki ga je opravil. V kolikor hoče, se lahko dejanja loti ponovno in tako naredi isto delo večkrat.

K ozaveščanju ostalih lahko pripomore tako, da svoje dejanje deli preko različnih socialnih medijev, multimedijskih ali elektronskih sporočil. Bistvo aplikacije je širjenje dobrih del in ozaveščanje ostalih; več kot bo sodelujočih, večji bo uspeh. Uporabniki morajo ostati vseskozi motivirani, za uporabljanje aplikacije, kar dosežemo z napredkom v dostopnem številu možnih dejanj.



Slika 3.1: Diagram primerov uporabe

Ko končamo nekaj dejanj, se nam odprejo nova, izmed katerih lahko izbiramo, katera bi opravljali. Naslednja stopnja napredka je, da ob določeni kvoti končanih dejanj začnemo opravljati dejanja s prijateljem. V sistem vpišemo prijateljev vzdevek in ko oba potrdiva, lahko dejanje začneva opravljati skupaj. Tako dodamo aplikaciji faktor mreženja in jo naredimo še za odtenek zabavnejšo.

Vedno nam je na voljo vpogled v statistiko, kjer vidimo, koliko dejanj smo že naredili in kako nam gre. Da je občutek povezanosti in napredka še večji, lahko sistem preko spletne povezave preveri, kako učinkoviti smo vsi uporabniki aplikacije. Število skupaj narejenih dobrih dejanj ugotovimo tako, da nam aplikacija vrne podatke, kolikokrat je bilo posamezno dejanje opravljeno s strani vseh uporabnikov. To v nas vzbudi občutek pripadnosti skupnosti. Hkrati nam aplikacija omogoča preverjanje napredka naših prijateljev, s katerimi smo že kdaj opravili kakšno dejanje. Če pri izpolnjevanju albuma naletimo na težavo, lahko izbermo pomoč, ki nam pomaga doumeti delovanje aplikacije. Pomoč je prav tako primerna za lažji začetek izpolnje-

vanja.

Naslednja stopnja napredka je dodajanje dejanj, katerih avtorji so uporabniki. Preko aplikacije se lahko povežemo na strežnik, od koder se pretočijo vsi predlogi uporabnikov za nove ideje in dejanja, ki so bili odobreni s strani administratorja. Tako lahko izberemo tista, ki se nam zdijo zanimiva, in ta se shranijo v naš album. Če si kasneje premislimo, lahko dejanja tudi izbrišemo.

Zadnja stopnja napredka je odobritev dostopa do pisanja in predlaganja idej za nova dejanja, ki bodo vidna in v navdih drugim uporabnikom. Diagram primerov uporabe je predstavljen na sliki 3.1.

Na tak način smo poskrbeli, da je uporabnik motiviran z napredkom in pripadnostjo. Še pomembnejša je notranja motivacija, ki jo v uporabniku vzbujajo dejanja sama in zavest, da pomaga ustvarjati boljši svet, v katerem hoče živeti [8].

3.2 Glavna aktivnost

Ob zagonu aplikacije se nam pojavi glavni meni. Gre za aktivnost, ki v isti datoteki vstavi tudi svoj fragment. Sam izgled menija je definiran v postavitvenih datotekah tipa XML. Tu imamo na zaslonu šest gumbov z napisi, kot je prikazano na sliki 3.2. Ob kliku na gumb se odpre nova aktivnost, ob kliku na izhod, se aplikacija zapre. Ob ustvarjanju nove aktivnosti najprej naložimo shranjene informacije o njenem preteklem stanju. Nato naložimo postavitveno datoteko, da se razporedijo elementi po ekranu. Sledi shranjevanje konteksta aplikacije v lokalno spremenljivko, ki nam bo prišla prav v nadaljevanju. Zadnji ukaz je prenos fragmenta v našo aktivnost. Sam fragment je definiran kot podrazred, ki razširja splošni razred Fragment. Znotraj definiramo konstruktor ter nekaj lokalnih spremenljivk. Nato prepišemo metodo, ki se kliče ob ustvarjanju fragmenta. Znotraj nje najprej pridobimo

Slika 3.2: Zaslonski posnetek glavne aktivnosti

ustrezno postavitveno datoteko. Spremenljivke nato povežemo z elementi postavitvene datoteke glede na njihova imena. Sedaj lahko vsakemu gumbu določimo svojega poslušalca, ki bo sprožil zanj predpisano dogajanje. Eden izmed gumbov aplikacijo zapre, ostali pa pokličejo novo aktivnost, ki jo uporabnik zahteva. To naredijo tako, da najprej ustvarijo nov objekt Namen (Intent). Določi se mu kontekst ter ciljni razred. Pripravljenega tako posredujejo naprej integrirani metodi za začetek aktivnosti, ki poskrbi za ustrezno razreševanje zahteve. Tako se obravnavana aktivnost ustavi in se prične izvajati nova. V razredu GlavnaAktivnost je določena tudi metoda *povezava()*. Aktivnosti jo kličejo pred dostopom do spleta, da preverijo, ali je povezava s spletom omogočena. V njej se najprej ustvari objekt *Upravljalec povezav*, nato se preveri, ali je možnost dostopati do spleta preko mobilnega omrežja. Sledi preverjanje dostopa preko brezžične povezave, na koncu se preveri ali je možen kakšen drug način aktivne povezave s spletom. V kolikor metoda ne najde nobene povezave, se izpiše sporočilo preko Androidovega priročnega sistema za sporočila *Toast* iz datoteke nizov, ki uporabnika obvesti, da mora

omogočiti povezavo s spletom.

Primer 3.1: Prikaz metode *povezava()*

```
@Override
protected void onCreate(Bundle savedInstanceState) {
public static boolean povezava() {
ConnectivityManager conM = (ConnectivityManager) ctx.
   getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo mobilni = conM.getNetworkInfo(
           ConnectivityManager.TYPE_MOBILE);
        if (mobilni != null && mobilni.isConnected()) {
           return true;
        }
        NetworkInfo wifi = conM.getNetworkInfo(
           ConnectivityManager.TYPE_WIFI);
        if (wifi != null && wifi.isConnected()) {
           return true;
        }
        NetworkInfo jeAktiven = conM.
           getActiveNetworkInfo();
        if (jeAktiven != null && jeAktiven.isConnected
           ()) {
return true;
        }
        Toast.makeText(ctx,R.string.niNeta, Toast.
           LENGTH_SHORT).show();
        return false;
}
```

Zadnja metoda v tem razredu je metoda *caka()*. V njej se ustvari podrazred Caka, ki se poveže s spletom. V skripto PHP posredujemo svojo identifikacijsko številko ter vzdevek. V odgovoru dobimo do deset čakajočih

Slika 3.3: Urejanje uporabnikovega vzdevka

zahtev za opravljanje dejanj, sestavljenih iz številke dejanja in vzdevka osebe, ki je dejanje predlagala. V kolikor imamo kakšno čakajočo zahtevo, se nam odpre nova aktivnost Cakajoce, ki poskrbi za izpis zahtev.

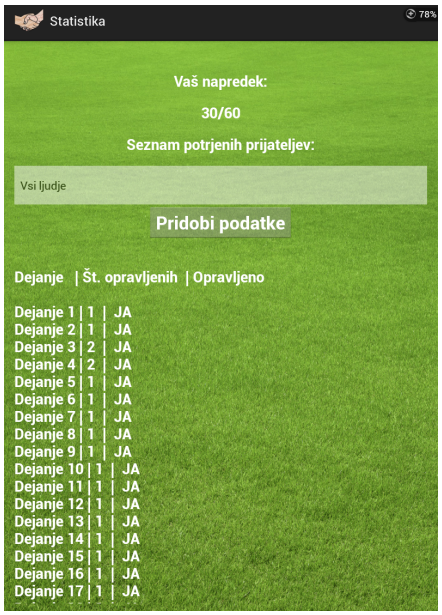
3.3 Nastavitve

Za popolno delovanje aplikacije potrebuje vsak uporabnik svoj vzdevek. Naloga aktivnosti Nastavitve je določanje vzdevka. Dolg je lahko do deset znakov in sestavljen iz nabora številk ter angleških črk. Ta omejitev je postavljena zaradi varnosti ter da imajo uporabnikovi prijatelji, ki ga želijo najti, lažje delo. Najprej se vnese željeni vzdevek, nato se aplikacija poveže z bazo na spletu in preveri, ali že obstaja nekdo z enakim vzdevkom. Če je vzdevek prost, ga uporabnik sme uporabiti. Nato se preveri, če uporabnik že obstaja v bazi s kakšnim drugačnim vzdevkom. To se zgodi na podlagi identifikacijske številke, ki se mu dodeli ob prvem vpisu v bazo. Številka se dodeli po

ključu, ki se avtomatično povečuje v bazi; če je bil zadnji vpisani uporabnik Miha z identifikacijsko številko 235, se bo novemu uporabniku Petru dodelila nova številka 236. Tudi če se Peter odloči, da spremeni svoj vzdevek v Peter-Veter, bo pri tem obdržal številko 236. Takšen način beleženja sem izbral, ker sem hotel uporabniku omogočiti spremembo svojega vzdevka, brez da bi pri beleženju v bazi na spletu izgubil svoj napredek. Prav tako je to zaščita pred večimi vnosi v bazo iste osebe, s čimer dobimo posledično točnejše podatke pri statistiki. Izgled same aktivnosti je prikazan na sliki 3.3.

Pri kodiranju sem najprej določil nekaj lokalnih spremenljivk za nadaljnjo uporabo. Nato sem v metodi, ki se zažene ob ustvarjanju aktivnosti, poklical ustrezno postavitveno datoteko. Tu se srečamo tudi s shranjevanjem podatkov v *skupne preference*. Tak način shranjevanja je primeren za manjše število podatkov in je ustrezen za obravnavano aplikacijo. Shranjuje se v formatu ključ-vrednost v notranji spomin naprave, podatki pa so dostopni le aplikaciji. Do teh vrednosti lahko dostopamo iz celotne aplikacije, kar se izkaže za zelo uporabno. Iz omenjenega skladišča naložimo v lokalno spremenljivko unikatno število uporabnika, če ta že obstaja (smo se že povezali na splet). Ostale lokalne spremenljivke povežemo z elementi iz postavitvene datoteke. Sledi nastavljanje vidljivosti gumbov za dodajanje, brisanje in potrjevanje spremembe vzdevka. Nato nastavimo poslušalce klikov na gumbe in odgovore na njih. Ko pritisnemo na gumb za brisanje, se spremeni vidljivost določenih gumbov, ponastavi se prostor, ki prikazuje trenutni vzdevek in tako ga v skupnem skladišču izbrišemo. Ob kliku na gumb za dodajanje spremenimo vidljivost nekaterih elementov in v ospredje pripeljemo tipkovnico, ki uporabniku omogoča vpisovanje vzdevka. Na koncu dodajanja potrdimo spremembo vzdevka. Ob tem se pokliče metoda *preveri()*, ki preko svojega podrazreda ugotovi prostost vzdevka na spletu in ga ustrezno vstavi. Podrazred uporabljamo zato, ker lahko izvajanje preselimo iz glavne niti delovanja. Povezava s spletom včasih traja nekoliko dlje, zato ne smemo dopustiti, da ostane glavna nit blokirana, saj lahko Android hitro ustavi delovanje aplikacije, v kolikor posumi, da se je izvajanje ustavilo [20].

Komunikacija s spletom poteka preko dveh metod tega podrazreda. V prvi pripravimo podatke in se povežemo, v drugi pa ustrezno obdelamo odgovor strežnika. Najprej iz skupnega skladišča pridobimo podatke o stanjih dejanj ter jih shranimo v spremenljivke. Pripravimo si tudi vzdevek, ki bi ga radi preverili in identifikacijsko številko uporabnika, če jo ima. Podatke nato shranimo preko dodatnih ključev v seznam, ki je sedaj pripravljen za nadaljnjo obdelavo. Dodamo tudi naslov, kamor naj se aplikacija poveže. Nato se podatki pošljejo na strežnik, metoda pa poskrbi za sprejetje podatkov in ustrezno obdelavo. Ob napaki pri sprejemanju ali pošiljanju podatkov se prikaže uporabniku informativno sporočilo. V drugi metodi objekt JSON, ki smo ga sprejeli kot rezultat komunikacije s strežnikom, razstavimo in shranimo sprejete vrednosti v ustrezne lokalne spremenljivke. V primeru, da je vzdevek zaseden, se uporabniku izpiše obvestilo, v nasprotnem primeru pa se shrani vzdevek tudi v skladišče na napravi, prav tako se skladno prilagodi tudi vidljivost elementov. Na strežniški strani skripta, ki čaka na podatke preko metode *HTML POST*, vzpostavi najprej povezavo s svojo bazo. Nato poslane podatke shrani v lokalne spremenljivke in se prepriča, da med poslanimi podatki ni zlonamerne kode. Kasneje preveri, ali v bazi že obstaja enak vzdevek. V primeru ujemanja se pošlje ustrezna vsebina v aplikacijo in izvajanje se zaključi. Če je vzdevek prost, skripta preveri, ali gre za prvi vpis uporabnika ali pa se je v preteklosti že vpisal. To preverimo s pomočjo identifikacijske številke. V prvem primeru se mu dodeli nova številka, ki bo vseskozi spremljala napravo oziroma osebo. Izvede se vstavljanje v bazo. V drugem primeru se uporabnikov zapis v bazi samo posodobi. Poleg vzdevka se v bazo shranijo tudi podatki o napredku uporabnika, torej koliko in katera dejanja je že končal. Vsi podatki v odgovoru se nato zakodirajo v objekt JSON in pošljejo v aplikacijo. Povezava z bazo se na koncu zaključi [7].



Slika 3.4: Zaslonski posnetek aktivnosti Statistika

3.4 Pomoč

V aktivnosti Pomoc se uporabnik seznani z delovanjem aplikacije. Zagotoviti sem hotel enostavno uporabo, a je pri dojemanju določenih možnosti aplikacije pomoč dobrodošla. Tu je obrazložen tudi sistem napredka, da se oseba zaveda potencialnih možnosti, ki se ji odpirajo. Programerski del je bil dokaj enostaven. Poklicati sem moral le postavitveno datoteko, v njej pa je preko datoteke, ki hrani vrednosti nizov, napisana vsebina celotne pomoči.

3.5 Statistika

Namen te aktivnosti je popestriti aplikacijo in uporabniku podati vpogled, koliko dobrega so naredili vsi uporabniki skupaj, torej kolikokrat je bilo posamezno dejanje opravljeno. Če je uporabnik opravljal kakšno dejanje tudi s prijateljem, se njegov vzdevek doda na seznam potrjenih prijateljev. Omogočen mu je vpogled v opravljena dejanja prijatelja. Vsebina se pretoči

s spleta s pritiskom na gumb *Pridobi podatke*, kot je razvidno iz slike 3.4.

V datoteki najprej določimo lokalne spremenljivke. Nato v metodi, ki se zažene ob ustvarjanju aktivnosti, najprej nastavimo povezavo do postavitvene datoteke. Lokalne spremenljivke povežemo z elementi, definiranimi v postavitveni datoteki. Shranimo tudi referenco do skupnega skladišča. Nato se z algoritmom sprehodimo čez vse vrednosti v skladišču, ki beležijo stanje napredka posameznega dejanja, in seštejemo vsa končana dejanja. Število prikažemo v ustreznem gradniku. Sledi ustvarjanje menija, iz katerega lahko izbiramo med prikazom statistike za vse uporabnike in za določenega prijatelja. Iz podatkovnega skladišča pridobimo potrjene prijatelje, ki jih dodamo na seznam, nato pa jih s pomočjo adapterja prikažemo uporabniku. Pripnemo poslušalec na gumb za pridobitev podatkov. Ob pritisku ločimo, ali si uporabnik želi pridobiti podatke o vseh uporabnikih ali o katerem izmed svojih prijateljev. V prvem primeru se pokliče metoda *osvezi()*, v drugem pa metoda *najdi()*. Metodi sta si podobni, a se razlikujeta v nekaterih podrobnostih. Metoda *osvezi()* vsebuje podrazred Osvezi, ki prenese delovanje iz glavne niti na vzporedno. V prvi metodi podrazreda pripravimo podatke o napredku posameznih dejanj, ki se bodo zapisali v bazo. Ustrezno jih shranimo v seznam parov ključ-vrednost ter jih pošljemo na strežnik z metodo *html POST*. Rezultat sprejmemo in ga posredujemo drugi metodi v podrazredu. Ta odgovor JSON spremeni v polje nizov. Nato se vsi nizi združijo v pregledno obliko, ki se izpiše uporabniku na zaslonu. V primeru napake pri pošiljanju ali sprejemanju podatkov na strežnik uporabnika obvestimo. Na strežniku podatke sprejme skripta PHP, ki najprej ustvari povezavo z bazo. Spremenljivke, sprejete iz metode HTML POST, shrani v svoje spremenljivke zaradi varnosti. Nato jih zapiše v bazo k ustreznemu uporabniku. V drugem delu skripta izvede iskanje po vseh zapisih uporabnikov in sešteje, koliko jih je končalo posamezno dejanje. Rezultat, zapakiran v objekt JSON, vrne aplikaciji kot odgovor. V metodi *najdi()* je delovanje podobno. V izpisu uporabniku se prikaže uspešnost le za iskanega uporabnika in za nas. Poizvedba v bazi je rahlo drugačna in zabeleži stanje opravljenih dejanj v

seznam, ki se pošlje s strežnika v aplikacijo. Tam se ustrezno pretvori v uporabniku prijazno besedilo in se izpiše.

3.6 Okvir albuma

Aktivnost AlbumOkvir služi kot okvir, v katerem se izmenjujejo strani albuma, torej fragmenti. Najprej določimo lokalne spremenljivke. V metodi, ki se pokliče ob ustvarjanju aktivnosti, najprej pokličemo postavitveno datoteko. Nato shranimo referenco na skladišče in iz njega pridobimo največje število opravljenih dejanj ter vzdevek uporabnika. Tega uporabimo za nastavitev naslova aktivnosti, ki bo viden skozi celoten album. Če vzdevek še ni nastavljen, se ne nastavi nič. Povežemo tudi spremenljivke z elementi iz postavitvene datoteke, da se na njih lahko sklicujemo. Preberemo, kako smo prišli v to aktivnost, in če smo prejeli zraven še dodatne parametre, se odzovemo tako, da v albumu odpremo zahtevano stran. Imamo tudi tri različne gumbe, ki s pritiskom na njih določijo, v kateri del albuma želi uporabnik preskočiti. Delujejo torej kot zavihki. Da jim lahko določimo funkcionalnost, jim moramo pripeti poslušalce. Napisana je tudi metoda *prestejDejanja()*, ki prešteje, koliko dejanj je v albumu opravljenih, in posodobi največje število teh, če je to potrebno. Vsebina okvirja, ki jo predstavlja ta aktivnost, se mora izmenjevati. Gleda na to, da uporabljamo koncept albuma dobrih dejanj, sem realiziral menjavanje vsebine, kot če bi listal album. Zaslon zazna poteg v levo in premakne vsebino na naslednjo stran, v primeru potega v desno pa prikaže prejšnjo stran v albumu. Ta funkcionalnost je narejena s pomočjo adapterja strani. Gre za samostojen razred, opisan v nadaljevanju. V metodi moramo poskrbeti, da napolnimo adapter s fragmenti, stranmi albuma, med katerimi se bo uporabnik lahko premikal. To naredimo v metodi *nastaviStrani()*, kjer ustvarimo seznam fragmentov. Dodamo le toliko strani, kolikor jih je uporabnik že odklenil. Ta podatek dobimo iz skladišča podatkov. Nato ustvarimo še nov objekt adapterja strani in oboje povežemo.

3.7 Adapter strani

Razred AdapterStrani nam omogoča izdelavo adapterja. V njem so definirane poleg konstruktorja tudi štiri metode, ki služijo za lažjo orientacijo po seznamu fragmentov.

Primer 3.2: Prikaz razreda AdapterStrani

```
public class AdapterStrani extends
   FragmentStatePagerAdapter {
        private List<Fragment> fragments;
        private Fragment mSedanjiFragment;
        public AdapterStrani(FragmentManager fm, List<
           Fragment> fragments) {
                super(fm);
                this.fragments = fragments;
        }
        @Override
        public Fragment getItem(int position){
                return this.fragments.get(position);
        }
        @Override
        public int getCount(){
                return this.fragments.size();
        }
        @Override
public void setPrimaryItem(ViewGroup container, int
   position, Object object) {
        if (sedanjiFragment() != object) {
                mSedanjiFragment = ((Fragment) object);
        }
        super.setPrimaryItem(container, position,
```

Slika 3.5: Zaslonski posnetek naslovne strani albuma

```
            object);
        }
        public Fragment sedanjiFragment() {
                return mSedanjiFragment;
        }
}
```

3.8 Naslovna stran v albumu

AlbumNaslovna je ime prvega fragmenta, ki se prikaže v albumu. Služi kot naslovna stran in poudari videz albuma. Prikaže se uporabnikov vzdevek ter njegov napredek. Slednji je predstavljen s številom pobarvanih zvezdic, kot je prikazano na sliki 3.5.

Postopek izdelave fragmenta je tukaj podoben. Na začetku se ustvarijo lokalne spremenljivke, razred se poveže s postavitveno datoteko, nato pa se

povežejo spremenljivke z elementi, definiranimi v postavitveni datoteki. Nastavimo lastnosti ocenjevalne lestvice, ki prikazuje napredek. Število zvezdic, ki ponazarjajo uspešnost, nastavimo na šest, saj je šest stopenj za odklep, ter dodamo omejitev, da uporabnik ne more vplivati na število zvezdic direktno s pritiskom na element. Sledi še pridobitev vzdevka iz skladišča, če vzdevek obstaja. Nazadnje pridobimo največje število opravljenih dejanj iz skladišča in pobarvamo ustrezne zvezdice glede na napredek.

3.9 Trije tipi fragmentov

V albumu obstajajo trije tipi fragmentov. Imajo podobne lastnosti, a se razlikujejo v nekaterih konceptih in načinih delovanja. Prvi tip predstavlja osnovno opravljanje dejanj, drugi tip je namenjen opravljanju dejanj s prijateljem, tretji tip pa predstavlja dejanja, ki so jih uporabniki dodali sami. Vsak tip ima več predstavnikov. V nadaljevanju bom predstavil tipe podrobneje.

3.9.1 Prvi tip fragmentov

Vsaka stran v albumu mora imeti svoj fragment. Strani je veliko, zato moram biti postopek dodajanja novih fragmentov enostaven. Dinamično se ne da povečevati njihovega števila, zato skopiramo kodo v vse nove fragmente prvega tipa in povečamo le prvo spremenljivko ST_FRAGMENT. Nanjo se nanašajo vse ostale uporabljene spremenljivke. Najprej se določijo spremenljivke za kasnejšo uporabo. V prvi metodi, ki se kliče ob ustvarjanju fragmenta, določimo pot do skladišča, kjer je shranjena slika tega fragmenta. Metoda, ki se kliče naslednja, je precej obsežnejša. Tu velja omeniti, v kakšnem stanju je lahko vsako dejanje. Možna stanja dejanj in razlage so podane spodaj, kot prikazuje slika 3.6.

- Stanje 0 – Dejanje ni začeto.
- Stanje 1 – Dejanje se izvaja.

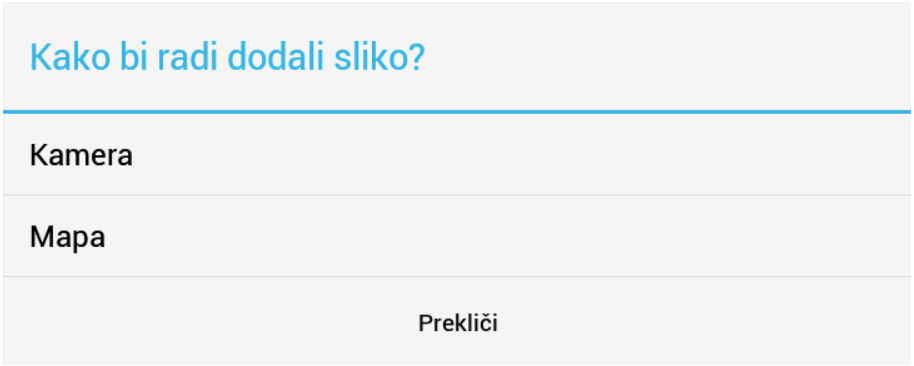
Slika 3.6: Prikaz različnih stanj dejanja

- Stanje 2 – Dejanje je končano.

Na začetku moramo ugotoviti, v katerem stanju je naše dejanje, da lahko temu primerno prilagodimo izgled strani. Pridobimo instanco skladišča, nato pa z njeno pomočjo poiščemo stanje dogodka in ga shranimo v spremenljivko STATUS_DOGODKA, ki je dosegljiva v celem fragmentu. Sledi kopica povezovanj elementov postavitvene datoteke s spremenljivkami. Vidljivost gradnikov se nastavi glede na stanje, prav tako se pokliče tudi ustrezno besedilo, ki je vidno uporabniku. Nato se nastavijo poslušalci pritiskov na gumb ter njihovi odzivi.

- Ob pritisku na gumb za začetek dejanja se status dogodka spremeni v 1 ter se shrani v skladišče. Tam se shrani tudi datum začetka opravljanja dejanja, prav tako se datum uporabniku prikaže. Gumb za začetek se skrije, pojavi pa se gumb za potrditev dejanja, preklic le-tega, gumb za dodajanj slik ter gumb za deljenje dogodka.

- Ob pritisku na gumb za opravljeno dejanje se status dogodka spremeni v 2 in se shrani na ustrezno mesto. Gumba za opravljeno dejanje in preklic dejanja se skrijeta, prikaže pa se gumb za ponovno opravljanje



Slika 3.7: Prikaz izbire za dodajanje slike

dejanja. Posodobi se tudi datum in napis, ki označuje, kdaj je bilo dejanje končano. Preveri se tudi, če je za dejanje nastavljena slika. V kolikor ni, se shrani privzeta slika, ki označuje zaključek dejanja.

- Pritisk na gumb, ki prekliče dejanje, ponastavi dejanje začetno stanje. Viden je le gumb za začetek dejanja. Status dejanja se spremeni v 0 in se shrani v skladišče. Prikaz datuma se skrije, iz notranjega spomina aplikacije se izbriše tudi slika, ki je nastavljena, na njeno mesto pa pride okvir, ki pričakuje sliko.
- Uporabnik lahko končano dejanje ponovno začne. Ukazi za dosego tega stanja so skoraj enaki kot pri gumbu za preklic dejanja. Ponastaviti je potrebno elemente, sliko, vrednosti v skladišču in izgled fragmenta.

Dodajanje slike

Pri dodajanju slike uporabnik pritisne na gumb za dodajanje. Prikaže se mu meni za izbiro vira slike. Doda jo lahko tako, da jo zajame s kamero naprave ali pa izbere fotografijo iz galerije. Prikaz izbire lahko vidimo na sliki 3.7. Izgradnja menija je narejena tako, da imamo poseben razred, v katerem je definiran izgled dialoga za izbiro. V njem se objektu določi naslov, seznam za izbiro vira slike, kaj storiti ob kliku na določen vir ter možnost preklica izbire. V metodi klicočega razreda sprejmemo rezultate in glede

Slika 3.8: Prikaz, kako se slika obrne za 90 stopinj

na uporabnikovo izbiro pokličemo novo aktivnost. Ta aktivnost vrne sliko. Sliko z metodo *zmanjsajSliko()* zmanjšamo na ustrezno velikost. S podanimi novi merami izberemo faktor zmanjšanja slike in delimo višino ter dolžino z izbranim faktorjem za ohranitev razmerja slike.

Primer 3.3: Prikaz metode *zmanjsajSliko()*

```
public Bitmap zmanjsajSliko(Bitmap bm, int novaSirina,
    int novaVisina) {
        int sirina = bm.getWidth();
        int visina = bm.getHeight();
        int faktor = Math.max(sirina/novaSirina, visina
            /novaVisina)+1;
        int fSirina = Math.round(((float) sirina) /
            faktor);
        int fVisina =  Math.round(((float) visina) /
            faktor);
```

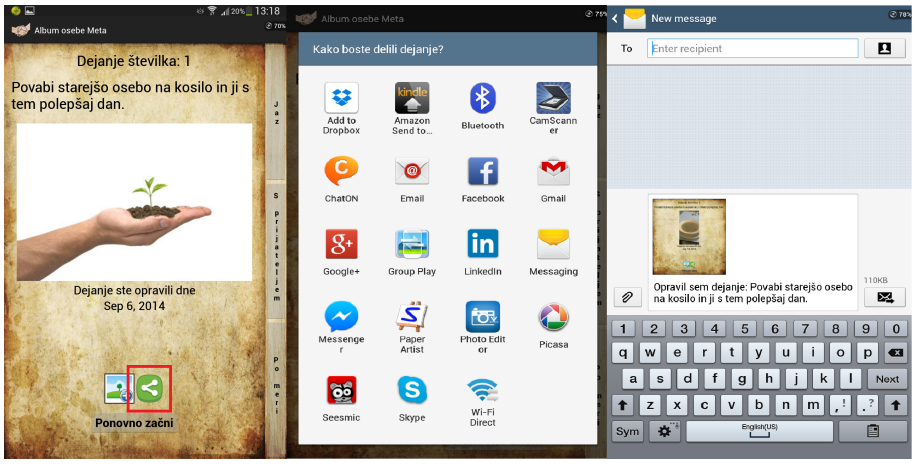
```
        Bitmap novaSlika = Bitmap.createScaledBitmap(bm
            , fSirina, fVisina, false);
        return novaSlika;
}
```

Omeniti je potrebno še tri metode. Prva shrani sliko v notranji spomin naprave. Najprej pridobimo instanco ovojnice razreda, s pomočjo katere dostopamo do datoteke, kjer se shranjujejo slike. Sliki nato dodamo ime ter jo shranimo. V primeru napake izpišemo sporočilo. Metoda vrne pot shranjene slike.
Druga metoda naloži sliko iz naprave. Podati ji je treba pot do shranjene fotografije, nato pa se ta naloži v predviden okvir.
S tretjo metodo lahko sliko obrnemo. Kadar fotografijo zajamemo s kamero telefona ali jo naložimo iz datoteke, se lahko zgodi, da je slika obrnjena drugače, kot bi si uporabnik želel. Rešitev tega problema je klik na sliko, pri čemer se obrne za 90 stopinj v smeri urinega kazalca. Implementacija tega je narejena tako, da naložimo sliko iz datoteke, kjer je shranjena. Nato jo pretvorimo v objekt, kasneje pa nad njim pokličemo matriko, obrnjeno za 90 stopinj. Novo nastalo sliko shranimo in jo naložimo uporabniku na vpogled. Postopek je prikazan na sliki 3.8.

Deljenje dejanja

Aplikacija uporabniku omogoča deljenje posameznih strani iz albuma s svetom. Za deljenje lahko uporabnik izbere katerokoli izmed nabora aplikacij, ki jih ima nameščene na napravi in mu to omogočajo, npr. preko Twitterja, Facebooka, preko elektronske pošte ali celo z multimedijskim sporočilom. Uporabnik zgolj klikne na gumb za deljenje. Pojavi se izbirno okno, ki mu omogoča izbiro željenega načina obveščanja prijateljev. Aplikacija pošlje sliko izbrane strani albuma, zraven doda še besedilo, ki dodatno pojasni, za katero dejanje gre. Deljenje je prikazano na sliki 3.9.

Ob pritisku gumba za deljenje se sproži metoda, ki najprej preveri, če



Slika 3.9: Postopek deljenja dejanja z drugimi

obstaja povezava s spletom, in nato pokliče metodo *pripraviZahtevo()*. Njen rezultat prikaže nove aktivnosti, ki nam omogočajo izbiro načina deljenja vsebine. V metodi *pripraviZahtevo()* najprej določimo vrsto zahteve. Pripeti moramo tudi sliko, ki predstavlja stran v albumu. To naredimo tako, da pridobimo instanco izbranega fragmenta, ki jo posredujemo metodi za zajem slike pogleda. Zahtevi dodamo še naslov oziroma temo, jo opremimo z dodatnim besedilom ter pripravimo sliko za deljenje. Na koncu vrnemo zahtevo. V tej metodi sta še dve, ki ju bom na kratko razložil.

Prva je imenovana *screenShot()*, ki kot atribut sprejme pogled, katerega sliko želimo zajeti. Problem nastane, ker nočemo narediti navadnega posnetka ekrana, ampak moramo zajeti celoten drseči pogled, tudi elemente, ki se jih zaradi premajhnega zaslona ne vidi. Rešitev je, da metodi podamo referenco na drseči element iz postavitvene datoteke. Na instanco platna zajamemo sliko podanega pogleda, ki smo ustrezno popravili dimenzije. Metoda vrne sliko pogleda.

Druga metoda iz podane slike in konteksta sliko shrani na napravo ter jo pripravi za pošiljanje. Zadnjo obravnavano metodo *prestejDejanja()* pokličemo, ko uporabnik konča dejanja. V njej preštejemo vsa opravljena dejanja in dobljeno število primerjamo z največjim številom opravljenih dejanj. To na-

redimo v izogib situaciji, kjer bi se uporabniku, ki želi ponovno opravljati že dokončano dejanje, dostop do odklenjenih dejanj zaklenil. Če gre za do sedaj največje število, to ustrezno zabeležimo v skladišču ter pokličemo novo aktivnost Odkleni in ji posredujemo ustrezen atribut za izpis uporabniku.

Primer 3.4: Prikaz metode *prestejDejanja()*

```
//Preveri, ce je uporabnik odklenil nova dejanja
public void prestejDejanja(){
        int skupaj=0;
        String stanje;
        SharedPreferences sh = getActivity().
           getSharedPreferences(getString(R.string.
           preference_kljuc), Context.MODE_PRIVATE);
        for(int i=1;i<=60;i++){
                stanje = "Stanje"+String.valueOf(i);
                stanje = sh.getString(stanje,"");
                if(stanje.equals("2")){
                        skupaj++;
                }
        }
        int max = sh.getInt("max", 1);
        if (skupaj > max) {
                max=skupaj;
                SharedPreferences.Editor editor = sh.
                   edit();
                editor.putInt("max", skupaj);
                editor.commit();
                Intent intent = new Intent(getActivity
                   (), Odkleni.class);
                switch (max){
                case 2:
                intent.putExtra("odkleni", 2);
```

```
                    startActivity(intent);
                    break;
        ...
```

3.9.2 Drugi tip fragmentov

Drugi tip fragmentov je namenjen opravljanju dejanj skupaj s prijateljem. Zasnovan je tako, da eden od uporabnikov vpiše prijateljev vzdevek, nato pa čaka na njegovo sprejetje. Prijatelju se naslednjič ob zagonu aplikacije prikaže sporočilo, da želi oseba, ki je dala pobudo, opraviti z njim določeno dejanje. Če se uporabnik strinja, se dejanje prestavi v fazo opravljanja. Prav tako se osebi medsebojno dodata na listo prijateljev namenjeno za opazovanje statistike. Ko končata, morata oba potrditi, da sta zaključila, in dejanje se smatra za opravljeno. Poleg vseh lastnosti, ki jih imajo fragmenti prvega tipa, je tu dodanih nekaj nadgradenj. Ob pritisku na gumb za začetek opravljanja dejanja se nam pojavi okno, ki od nas zahteva, da vpišemo prijateljev vzdevek, s katerim bi radi opravljali dejanje. Če imamo povezavo s spletom in nismo vnesli svojega vzdevka, lahko nadaljujemo in pokliče se metoda *rezerviraj()*.
Ta vsebuje podrazred Rezerviraj, ki se na ločeni niti poveže s skripto PHP za dostop do baze. Strežniku se pošlje stanje opravljenih dejanj in ta jih shrani v bazo. Če obstaja iskani prijatelj v bazi, lahko z odgovorom strežnika shranimo podatke o napredku na napravo, kot smo to storili v fragmentih prvega tipa. Z metodo *dodajPrijatelja()* se potrjeni prijatelj doda na seznam prijateljev, ki se ga uporablja pri statističnem poizvedovanju. Podobna metoda, kot je *rezerviraj()*, je metoda *naredi()*. Uporabljamo jo pri potrjevanju dejanja, ko hočeta uporabnika zaključiti dejanje. V nadaljevanju je predstavljena skripta PHP na strežniški strani za metodo *rezerviraj()*.

Primer 3.5: Prikaz skripte PHP na strežniku za rezervacijo osebe

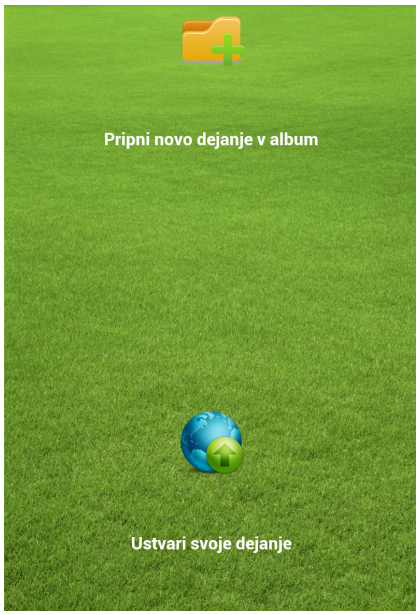
```
//Faza rezerviraj
```

```
$nik = mysqli_fetch_array(mysqli_query($con,"SELECT *
   FROM napredek WHERE vzdevek='$prijatelj';"));
$ena["pr"] = $nik['vzdevek'];
if ($nik[$mesto] == $nickname || $nik[$mesto] == 1){
  $sql="UPDATE napredek
  SET $mesto='1'
  WHERE id='$id';";
  $ena["pr1"] = 1;
  if (!mysqli_query($con,$sql)) {
    die('Error: ' . mysqli_error($con));
  }
}
else{
  $ena["pr1"] = $nik[$mesto];
}
```

Na strežniku najprej ustvarimo povezavo z bazo. Nato ugotovimo, kateri fragment se hoče povezati z bazo, in pridobimo mesto, kamor bomo shranjevali podatke v tabeli. Sledi pretvarjanje sprejetih podatkov v spremenljivke, s katerimi bomo operirali. Preverimo tudi, ali prijateljev vzdevek, s katerim želimo opraviti neko dejanje, sploh obstaja. Če obstaja, se izvajanje nadaljuje, v nasprotnem primeru skočimo na konec skripte. Nato se posodobijo vsa dejanja v tabeli, da dobimo aktualno sliko končanih dejanj. Na mesto, določeno s podano številko fragmenta, vstavimo vzdevek prijatelja, s katerim bi radi opravili dejanje.

3.9.3 Tretji tip fragmentov

Tretji tip fragmentov je narejen tako, da lahko uporabniki sami dodajo dejanja, ki se jim zdijo vredna opravljanja, izmed množice dejanj, ki so jih ustvarili ostali uporabniki. Ob določenem številu opravljenih dejanj se uporabniku odpre možnost dodajanja teh del v svoj album. Po dodatnem napredovanju



Slika 3.10: Prikaz dveh izbir za dodatna dejanja

lahko svoje ideje za dejanja tudi sami naložijo v bazo dejanj.
Tudi tu so izhodišče dejanja prvega tipa. Naloga posameznega dejanja se prebere iz skladišča, kamor se predhodno shrani v aktivnosti IzberiDejanje. Če želimo dejanje izbrisati, da bi imeli prostor za nova dejanja, uporabimo gumb za izbris dejanja.

3.10 Dodatna dejanja

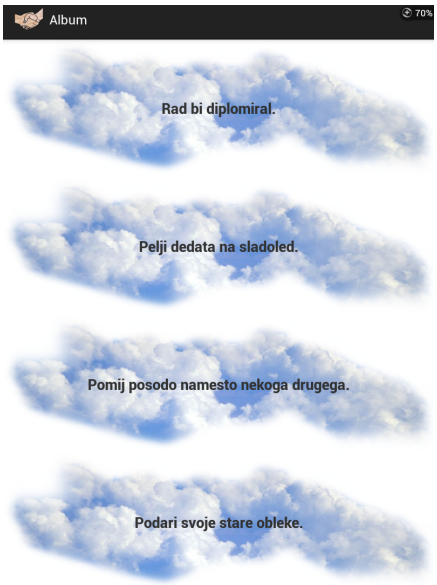
V aktivnosti Dodatno uporabnik izbira med dvema možnostma. Za obe potrebuje določeno število predhodno opravljenih dejanj, v nasprotnem primeru aplikacija prikaže sporočilo, da dejanja še niso odklenjena. Za lažjo predstavo nam služi slika 3.10. Prva je možnost, da uporabnik predlaga dejanje, dolgo do 160 znakov, ki bo po potrditvi administratorja vidno vsem uporabnikom in ga bodo lahko vstavili v svoj album. V aplikaciji je to uresničeno z oknom, ki uporabnika vzpodbudi, da naj vnese svojo idejo, kot je prikazano na sliki 3.11. Preveri se povezava s spletom, nato pa se dejanje z metodo *vstavi_dejanje()*

Slika 3.11: Predlaganje svojega dejanja uporabnikom

shrani v bazo dejanj na splet. Strežniška stran sprejme opis dejanja, vzdevek avtorja ter doda zraven še status neodobrenega dejanja. Dejanje se nato shrani v bazo, kjer čaka na odobritev administratorja. Uporabniku strežnik odgovori s potrdilom o uspešnosti vstavljanja dejanja. Drugo dejanje, ki ga lahko izbere uporabnik, je pritisk na gumb za dodajanje dejanj v album. Tu se ob pritisku ustvari zahteva in izvajanje programa se nadaljuje z aktivnostjo IzberiDejanje.

3.11 Izbiranje dejanja uporabnikov

V aktivnosti IzberiDejanje so zbrana vsa dejanja, ki so jih predlagali uporabniki in odobrili administratorji. Izvede se poizvedba na strežnik, ta pa vrne seznam odobrenih dejanj, ki se prikažejo uporabniku. Seznam je prikazan na sliki 3.12. Ob pritisku na eno izmed dejanj, se to vstavi v album. Če so vsa mesta zasedena, aplikacija uporabnika o tem obvesti in mu predlaga izbris odvečnih dejanj. Aktivnost najprej pokliče metodo *pokazi()*. Ta v podra-



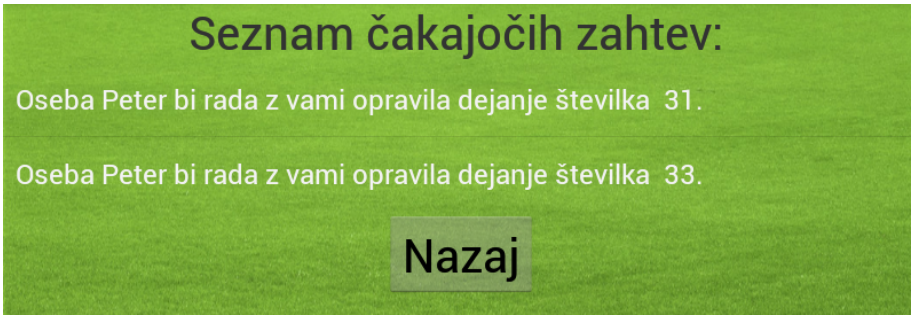
Slika 3.12: Primer seznama dejanj, ki so jih predlagali uporabniki

zredu Pokazi dostopa v vzporedni niti do strežnika in baze. Strežnik vrne seznam dejanj aplikaciji v kateri se izpišejo uporabniku. Vsakemu dejanju nastavi tudi poslušalec, tako da se ob pritisku na željeno dejanje to prenese v album na svoje mesto. Odpre se nam album na tisti strani, kjer smo vstavili dejanje. Če je album poln, se nam izpiše ustrezno sporočilo. Hkrati se zažene nit, ki traja toliko časa, da se sporočilo prikaže in umakne. Nato se odpre album na strani, kjer bi lahko izbrisali dejanja.

3.12 Odklepanje novosti

Namen aktivnosti Odkleni je, da ko uporabnik konča določeno število dejanj in se mu omogoči opravljanje novih, uporabnika o tem obvesti, kot je prikazano na sliki 3.13. Ob različnem številu narejenih dejanj se prikažejo drugačna sporočila. Po zaprtju tega sporočila se uporabnika preusmeri na novo odklenjena dejanja. Najprej pripnemo aktivnosti njeno postavitveno datoteko ter elementom iz nje dodamo funkcionalnost. Nato preberemo do-

Slika 3.13: Izpis uporabniku, ko odklene nova dejanja.



Slika 3.14: Prikaz sporočila, da hoče nekdo opravljati dejanje z uporabnikom

daten parameter zahteve, ki je začela obravnavano aktivnost. Glede na vrednost parametra izpišemo uporabniku sporočilo. Prepišemo tudi metodo za pritisk gumba nazaj na napravi, da uporabnika pošlje v album na ustrezno stran.

3.13 Čakajoče zahteve

Aktivnost Cakajoce nam prikaže čakajoča sporočila o uporabnikih, ki bi radi opravljali dejanje z nami. Prikaz je na sliki 3.14. Če ni čakajočih dejanj, se ne prikaže nič. V nasprotnem primeru se izpiše do deset dejanj, ki nas čakajo. Vidimo lahko vzdevek uporabnika, ki želi opravljati dejanje z nami ter številko dejanja, ki ga želi opraviti. Ob pritisku na sporočilo nas aplika-

cija preusmeri na stran v albumu, ki ustreza iskanemu dejanju. Na začetku povežemo postavitveno datoteko in elemente v njej z aktivnostjo ter sprejmemo podatke iz zahteve, ki je poklicala našo aktivnost. Gre za polje nizov, ki smo jih predhodno pretočili s spleta. Sestavljeni so iz vzdevkov in števil dejanj, ki jih obdamo z besedilom Nato jih dodamo v nov seznam stavkov. Uporabniku izpišemo seznam in dodamo poslušalce elementom s seznama. Ob pritisku ustvarimo zahtevo za novo aktivnost, da se odpre stran v albumu. Vredno je omeniti še strežniško stran pridobivanja podatkov o čakajočih zahtevah. Najprej se vzpostavi povezava z bazo in se prebere sprejeta podatka o identifikacijski številki in vzdevku uporabnika, za katerega poizvedujemo. Nato preverimo po celotni bazi, ali je kje zapisano njegovo ime, da bi z njim začeli opravljati dejanje. Na koncu vrnemo seznam desetih vzdevkov in dejanj, če pa jih je manj, vrnemo ničle. V aplikaciji se nato izvede obdelava sprejetih podatkov.

Primer 3.6: Prikaz dela skripte PHP za ustvarjanje seznama čakajočih dejanj

```
$result = mysqli_query($con,"SELECT * FROM napredek");
while($row = mysqli_fetch_array($result)) {
        if ($row['id']!=$id && $i < 10){
                if ($row['tprvi']==$nickname){
                        $rezultat[$i] = 31;
                        $rezultat[$i+10] = row['
                           vzdevek'];
                        $i++;
                }
        ...
```

3.14 Seznam dejanj

Seznam dejanj je nastal na podlagi pogovorov s prijatelji, lastnih idej in zapisov drugih [3, 19].

1. Povabi starejšo osebo na kosilo in ji s tem polepšaj dan.
2. Danes podari 5 iskrenih komplimentov.
3. Najdi staro sliko s prijateljem in ga razveseli.
4. Preseneti mamo/partnerico/prijateljico z rožo.
5. En teden zavestno recikliraj.
6. Spusti nekoga pred seboj v vrsti.
7. Naredi nekaj dobrega za nekoga, ki ne bo nikoli izvedel.
8. Nauči se rojstne dneve bližnjih.
9. Pohvali otroke, ki jih srečaš.
10. 3 dni ne reci ničesar slabega o nikomer.
11. Prevzemi odgovornost za nekaj, kar si krivil druge.
12. Dva dni ozaveščaj ljudi o pomembnosti medsebojne pomoči.
13. Podari obleke iz omare, ki jih več ne potrebuješ.
14. Deli svoje znanje z nekom, ki ga potrebuje (inštrukcije, nasveti).
15. Podari nekomu svojo knjigo.
16. V šolo ali službo prinesi piškote in jih deli s sošolci ali sodelavci.
17. Soseda povabi na pijačo.
18. Daj osebi, nad katero si obupal, še eno možnost.
19. Osveži osnovno znanje oživljanja. Morda nekomu rešiš življenje.
20. Zapomni si imena naslednjih 5 ljudi, ki jih spoznaš.
21. Nauči se 3 stvari, ki jih ceniš pri starejših.

22. Razmisli o možnosti, da postaneš darovalec organov.
23. Pohvali kuharja, če si užival v obroku.
24. Glasbenikom na ulici podari nekaj kovancev.
25. Pelji na sprehod psa, čigar lastnik nima časa.
26. Pojdi namesto nekoga v trgovino.
27. Namesto z osornostjo odreagiraj z nasmehom. 3x.
28. Od vsakega človeka, ki ti je blizu, se nekaj nauči. Naredi seznam.
29. V naslednjih treh pogovorih, ki jih boš imel, postani najboljši poslušalec.
30. Na svoj rojstni dan posadi drevo.
31. S prijateljem pojdi darovati kri.
32. Namesto kosila podarita s prijateljem obrok lačnemu.
33. Naredi dve ptičji hranilnici s prijateljem.
34. S prijateljem naredita malo zabavo, kjer bo veliko smeha.
35. S prijateljem opustita eno izmed svojih slabih navad za en teden.
36. Izmenjaj knjigo z navdihujočo tematiko s prijateljem.
37. S prijateljem ustvarita tekmovanje, kdo zbere več zamaškov, in jih podarita.
38. V stilu namiznega tenisa si s prijateljem en teden izmenjujta opravljena dobra dejanja.
39. S prijateljem naredi nekaj dobrega za otroke iz soseske.
40. Ti in tvoj prijatelj pošljita zahvalno pismo nekomu, ki si ga zasluži.

41. Preberi si, kaj je to altruizem. Prebrano deli s prijateljem.
42. Nauči prijatelja nekaj skuhati, on pa tebe.
43. Izkoristi dan in pojdi s prijateljem na izlet.
44. S prijateljem zberita stare igrače in jih podarita.
45. Združita moči s prijateljem in nekomu nekaj popravita.
46. Naberi nekaj sadja ali zelenjave s prijateljem in ga razdeli.
47. S prijateljem izberita si dan in tisti dan delita objeme.
48. Združita moči s prijateljem in nekomu nekaj ustvarita.
49. Čas za nagrado. Naredi nekaj lepega za prijatelja, on pa zate.
50. S prijateljem naredi nov seznam dobrih dejanj, kajti ta bo počasi poln.

3.15 Baza podatkov

Na strežniku se nahaja baza podatkov. Aplikacija komunicira s podatki v bazi preko skript PHP. V bazi imamo podatke spravljene v dveh tabelah. Prva se imenuje Napredek in hrani podatke o uporabnikih in stanjih njihovih dejanj. Druga tabela se imenuje Dejanja in vsebuje množico dejanj, ki so jih uporabniki dodali.

3.15.1 Tabela Napredek

Tabela Napredek je sestavljena iz 62 stolpcev. V prvem stolpcu se hrani indeks, ki predstavlja vsako napravo, ki dostopa do tabele. Vsak nov uporabnik dobi svojo identifikacijsko številko, ki ob spremembi vzdevka ostane enaka. Drugi stolpec hrani vzdevke uporabnikov in tudi te se ne smejo podvojevati. Preko njih se lahko dostopa do opravljanja dejanj s prijatelji. Ostalih šestdeset stolpcev predstavlja vsako dejanje, v katerem so lahko vrednosti 0, 1, 2 ali vzdevek prijatelja.

Indeks	Vzdevek	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
29	Edita	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	Matic	0	1	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	Peter	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	Test	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	Test2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	Test3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Slika 3.15: Prikaz shranjenih zapisov v tabeli Napredek

Indeks	Vzdevek	Odobreno	Opis Dejanja
1	Matic	1	Podari svoje stare obleke.
2	Petra	0	Pomij posodo namesto nekoga drugega.
13	Edita	0	Pelji dedata na sladoled.
15	Matic	0	Rad bi diplomiral.

Slika 3.16: Prikaz shranjenih zapisov v tabeli Dejanja

3.15.2 Tabela Dejanja

Tabela Dejanja je sestavljena iz štirih stolpcev. Prvi stolpec predstavlja identifikacijsko številko, ki se z vsakim dodanim dejanjem avtomatično povečuje. Drugo polje hrani opis dejanja, predlaganega s strani uporabnika. V tretjem stolpcu so shranjeni vzdevki uporabnikov, ki so avtorji dejanj. V zadnjem stolpcu je status odobrenega dejanja predstavljen s številom 1, če dejanje še ni odobreno, je shranjena vrednost nastavljena na 0.

3.15.3 Izpis podatkov iz tabel

Na spletu lahko preverimo vnesene elemente v bazi. Z dvema različnima skriptama PHP dostopamo do obeh tabel, iz katerih pridobimo podatke in jih izpišemo. Primer izpisa za tabelo Napredek je na sliki 3.15 , za tabelo Dejanja pa na sliki 3.16.

3.16 Manifest aplikacije

Gre za glavno datoteko programa, ki je v vsaki aplikaciji obvezna. Operacijski sistem ob zagonu programa najprej obdela to datoteko in iz nje razbere informacije o aplikaciji. Določi se ime paketa Java. Zapisana so dovoljenja, ki jih potrebuje aplikacija za delovanje, na katerih različicah Androida deluje, katere velikosti ekranov podpira, zahteve telefona, ki jih potrebuje aplikacija za delovanje ter potrebne podporne knjižnice. Določeno je tudi ime aplikacije, začetna ikona, tema aplikacije ter aktivnosti, ki jih zajema [1, 12].

3.17 Zunanji viri

Splošno pravilo programiranja nam veleva, da je dobro ločiti vire, ki jih uporabljamo v kodi, od same kode. S tem je omogočeno lažje vzdrževanje, upravljanje in posodabljanje virov. Pri programiranju za operacijski sistem Android je ločevanje virov dodobra podprto. V nadaljevanju je predstavljenih več tipov virov [5].

3.17.1 Nizi

Vsa besedila in napisi so shranjeni v svoji datoteki. Ekipa Androida je s tem zelo poenostavila prilagajanje aplikacije za globalni trg, saj imamo na enem mestu zbrano vse potrebno za prevajanje. Kadar imamo na primer aplikacijo za slovenski in španski trg, namesto da shranimo datoteko z nizi v mapo *values*, naredimo mapi *values-es* in *values-si*, nato pa shranimo prevedeni datoteki z nizi vsako v svojo mapo. Uporabnik bo ob zagonu aplikacije avtomatično dobil ustrezno jezikovno različico glede na njegove lokalne nastavitve jezika. Končnice na koncu map *values* ustrezno določimo glede na kodo jezika [13].

3.17.2 Dimenzije

Za določanje razdalj imamo na voljo več mer. Najpogostejša in zaželena je uporaba mere *dp* (density-indipendant pixels). Gre za mero, kjer lahko določimo razdalje in velikosti elementov neodvisno od resolucije naprave. Za osnovno primerjavo velja, da je 1 piksel enak eni enoti dp pri ekranu velikosti 160 *dpi* (točk v enem inču). Prav tako lahko uporabljamo milimetre z oznako *mm*, inče z oznako *in*, *px* je oznaka za piksle, eno dvainsedemdesetino inča predstavlja enota *pt*, uporabljamo pa lahko tudi mero *sp* (scale-independent pixels) [14]. Določene mere, ki se uporabljajo večkrat, sem uporabil v svoji aplikaciji za določanje odmika elementov od roba ekrana. Te mere so shranjene v datoteki *dimens.xml*, podatki pa so v njih shranjeni in klicani podobno kot elementi datoteke z nizi.

Primer 3.7: Prikaz nastavljanja dimenzij v zunanjo datoteko

```
<resources>
    <dimen name="activity_horizontal_margin">16dp</
        dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
</resources>
```

3.17.3 Vir slike

V mapo *Viri* lahko dodamo svoje datoteke, kot je v našem *vir_slike.xml* primeru. Tam so shranjene vrednosti za polje nizov, ki se jih uporablja pri ustvarjanju možnosti, kako želi uporabnik dodati sliko k izbranemu dejanju.

Primer 3.8: Prikaz ustvarjanja svojega polja nizov v zunanji datoteki

```
<resources>
        <string-array name="vir_slike">
                <item>Kamera</item>
                <item>Mapa</item>
        </string-array>
```

```
</resources>
```

3.18 Postavitev elementov

Mapa, ki določa postavitev, je najpomembnejša mapa izmed vseh virov. V njej so shranjene datoteke tipa XML. Tam je definiran izgled in postavitev elementov v aplikaciji. Elementi so hierarhično urejeni, njihovi atributi pa določajo izgled določenega elementa. V nadaljevanju bom opisal uporabljene gradnike in njihove atribute.

3.18.1 Skupni atributi

Vsakemu gradniku moramo določiti atribute. Najprej se bom osredotočil na atribute, skupne večini gradnikov-

- Na prvem mestu je atribut *id*. Določa ga niz, preko katerega se lahko dostopa do samega gradnika.
- Sledita mu določanje širine in višine elementa. Določimo mu lahko natančno vrednost z eno izmed merskih enot, o katerih je bilo več povedanega v poglavju o dimenzijah. Bolj zaželena je uporaba relativnih mer. Prva nam ovije vsebino gradnika, medtem ko druga zapolni ves prostor svojega nadrejenega elementa.
- Zadnja skupna stvar je atribut ozadje. V kolikor želimo, da ima naš gradnik ozadje, mu ga določimo.

3.18.2 Gradniki

Na tem mestu so podrobneje razloženi gradniki, ki so bili uporabljeni v obravnavani aplikaciji.

- Drsni pogled (ScrollView) – Zaradi prilagajanja vsebine različno velikim napravam je pogosto potrebno uporabiti ta gradnik. Omogoča

nam, da lahko postavljamo vsebino izven trenutno vidnega polja ekrana, do nje pa dostopamo z drsenjem po ekranu. Kadar predstavlja prvi element, mu moramo kot atribut določiti shemo XML.

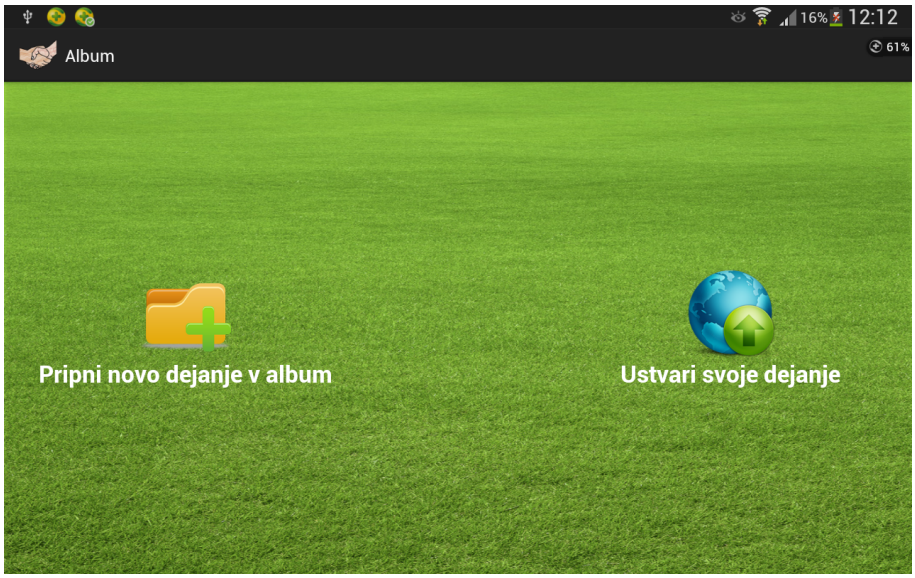
- Linearna postavitev (LinearLayout) – Gre za način gnezdenja podrejenih elementov, ki si sledijo zaporedno. Ločimo ga glede na orientacijo, vodoravno ali navpično, kar določimo z atributom. Dodamo lahko tudi atribut za odmaknjenost podedovanih elementov od robov. Posebnost linearne postavitve je atribut utež. Vsem podedovanim elementom se določi utež in vsak izmed njih zavzame svoj dolžinski delež glede na vsoto vseh uteži.

- Relativna postavitev (RelativeLayout) - Gre za način gnezdenja podrejenih elementov, vendar se namesto zaporedno, gradniki postavljajo relativno glede na njihovega starša.

- Gumb (Button) – Sestavljajo ga pogosti atributi. Prav tako mu lahko določimo tekst, barvo teksta in ozadje. Tu velja izpostaviti trik, ki se ga da uporabiti, da namesto zgolj besedila dodamo nad njim še sliko. To naredimo tako, da nastavimo ozadje gumba prozorno. Nato dodamo sliko nad besedilo.

- Pogled besedila (TextView) – Poleg običajnih atributov velja izpostaviti vse lastnosti, ki jih lahko nastavimo besedilu, npr. vir besedila, barvo, stil in velikost pisave.

- Urejevalno besedilo (EditText) – Ima vse standardne atribute. Namenjen je sprejemanju vnesenega besedila uporabnikov, zato ima tudi svoje posebne lastnosti. V obravnavani aplikaciji sem uporabil atribut, ki določa, da se vsak stavek začne z veliko začetnico, in s tem poenostavi uporabniški vnos. Omejil sem velikost vnesenega besedila na deset črk in določil, da lahko gradnik sprejme določene znake, ter predpisal katere. Tako je tudi varnost na višjem nivoju in se odstrani možnost vstavljanja zlonamerne kode.

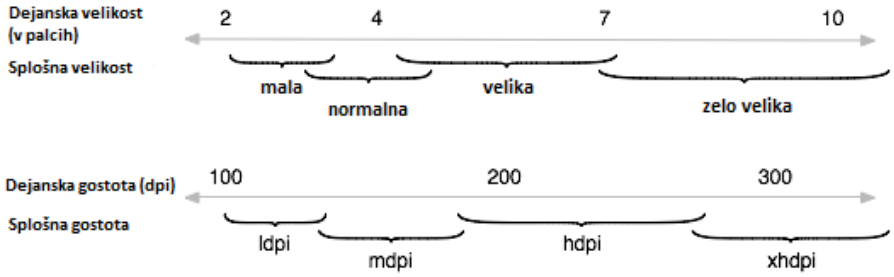


Slika 3.17: Primer ležeče postavitve elementov

- Pogled slike (ImageView) – Poleg id-ja, širine, višine, obrob in položaja, lahko temu gradniku nastavimo tudi vir slike, ki jo prikazuje. Vsebino lahko kasneje programsko zamenjamo. Prav tako lahko nastavimo opis slike in ali gradnik sprejema klike.
- Pogled strani (ViewPager) – Gre za okvir, v katerem se menjujejo fragmenti.
- Ocenjevalna lestvica (RatingBar) – Z osnovnimi atributi določimo njen izgled, nato pa prilagajamo njen izgled programersko. Nastavimo lahko število vseh vrednosti, prikažemo naš napredek in določimo, če uporabnik lahko spreminja vrednosti ročno.

3.19 Ležeča postavitev elementov

Pri ležeči postavitvi elementov lahko definiramo datoteke z enakimi imeni kot v mapi Postavitev. Razporeditev lahko prilagodimo položaju, ko uporabnik zasuče svojo napravo v vodoravno lego. Prikaz vidimo na sliki 3.17.



Slika 3.18: Prikaz, kako Android približno umesti dejanske velikosti in gostote pik v splošne velikosti in gostote pik (vrednosti so približne) [17].

Aplikacija samodejno zazna spremembo orientacije in nato izbere ustrezno datoteko. Če imamo definirano postavitev le za enojno orientacijo, se elementi postavijo enako v obeh primerih, le izgled je lahko nekoliko popačen. Lažja izbira je zaklep zaslona na pokončno orientacijo.

3.20 Grafični elementi

Slike, ikone in ozadja so razvrščeni v več map. Datoteke so v vseh mapah poimenovane enako, poimenovanje map pa se loči glede na kvaliteto slikovnega materiala. Groba razdelitev je prikazana na sliki 3.18. Njihova imena se končajo z eno izmed sledečih možnosti: ldpi, mdpi, hdpi, xhdpi, xxhdpi, xxxhdpi. Datoteke v njih so enake, vendar z različno resolucijo, da lahko Android izbere najprimernejšo za izbrano napravo. Za razmerje med velikostmi velja 2:3:4:6:8, kjer ldpi zaradi zastarelosti ni upoštevan [14].

Poglavje 4

Sklepne ugotovitve

V diplomskem delu smo predstavili razvito aplikacijo za širjenje dobrih del. Opisali smo uporabljene tehnologije, metodologijo in način razvoja. Predstavili smo delovanje ter pomembnejše koncepte.

Naredili smo koncept albuma, kjer se uporabnik premika po straneh s potegi po zaslonu. Aplikacija komunicira s spletom in podatke shranjuje v bazo na strežniku. Fotografije in ostale podatke shranjuje lokalno na napravi. Mreženje je podprto tako, da lahko dejanje opravljamo s prijateljem, izberemo dejanja, ki so jih predlagali drugi, sami predlagamo nova dejanja ter opravljena dejanja delimo s prijatelji. Omogočen je tudi pregled statistike opravljanja dejanj nas in ostalih.

Med razvijanjem so se pojavili tudi problemi. Dodatno delo je predstavljala uporaba knjižnic za podporo starejših različic Androida. Emulatorji raznovrstnih različic operacijskega sistema so omejeni s hitrostjo in funkcionalnostjo, kar je otežilo testiranja. Na koncu je bila večina problemov razrešenih, ostali pa so bili rešeni tako, da smo jih zaobšli.

V prihodnosti bi bilo smiselno aplikacijo objaviti na mobilnem trgu *Google Play*. Dobili bi povratno informacijo večih uporabnikov, ki bi me opozorili

na morebitne vrzeli. Izdali bi lahko nove različice in naredili aplikacijo robustnejšo.

Korak naprej bi bil, da se razvije skupnost, ki bi se širila in pridobivala nove člane v večjem obsegu. Posledično bi bilo tudi več možnosti za opravljanja dejanj in število idej bi se znatno povečalo. Uporabnikom bi lahko ponudili možnost ustvarjanja profila na spletni strani, kjer bi se lahko računa združila.

Skupnost bi lahko ustvarjala novice, ki bi jih lahko videli vsi. Opravljena dejanja bi se interaktivno prikazovala na zidu novic, kar bi bilo v vzpodbudo ostalim uporabnikom.

Literatura

[1] L. Dacey, S. Conder. *Sams Teach Yourself Android Application Development in 24 hours*, Sams, United States of America, 2010, str. 77-91.

[2] J. Heidemann et al. *Online social networks: A survey of a global phenomenon, Computer Networks, vol. 56, str. 3866–3878, 2012.*

[3] H. Jackson Brown Jr. Drobna navodila za življenje, Mladinska knjiga založba, Ljubljana, 2008.

[4] M. Krisper. Agilni pristopi k razvoju informacijskih sistemov, prosojnice s predavanj, Univerza v Ljubljani, Ljubljana, 15.6.2011.

[5] R. Meier. Professional Android Application Development, Wiley Publishing, Indianapolis, 2009, str. 52-65.

[6] D. Merrick. Six tips to building a huge social network, PCWorld, BrandPost, 15.7.2014.

[7] M. Štrancar, S. Klemen. Php in MySql na spletnem strežniku Apache, Pasadena, Ljubljana, 2005, str. 105-110.

[8] K. Werbach, D. Hunter. For the Win: How Game Thinking Can Revolutionize Your Business, Wharton Digital Press, Philadelphia, 2012.

[9] (2014) Wikipedia about Android operating system. Dostopno na: http://en.wikipedia.org/wiki/Android_%28operating_system%29#Features

[10] (2014) Dashboard about Android. Dostopno na: http://developer.android.com/about/dashboards/index.html

[11] (2014) Android activity life cycle. Dostopno na: http://developer.android.com/training/basics/activity-lifecycle/starting.html

[12] (2014) Application manifest file. Dostopno na: http://developer.android.com/guide/topics/manifest/manifest-intro.html

[13] (2014) How to support languages in Android. Dostopno na: http://developer.android.com/training/basics/supporting-devices/languages.html

[14] (2014) How to externalize resources. Dostopno na: http://developer.android.com/guide/topics/resources/more-resources.html

[15] (2014) Iconography in Android. Dostopno na: http://developer.android.com/design/style/iconography.html

[16] (2014) Getting the Android SDK. Dostopno na: http://developer.android.com/sdk/index.html

[17] (2014) How to support different screens in Android. Dostopno na: http://developer.android.com/guide/practices/screens_support.html

[18] (2012) O razvojnem okolju Android in njegova namestitev. Dostopno na: http://android.fri.uni-lj.si/index.php/Razvojno_Okolje

[19] (2014) Webpage hosting ideas and campaigns for a better world. Dostopno na: https://www.dosomething.org/

[20] (2014) Processes and threads in Android. Dostopno na: http://developer.android.com/guide/components/processes-and-threads.html

[21] (2014) About phpMyAdmin. Dostopno na: http://www.phpmyadmin.net/home_page/index.php

[22] (2014) PHP homepage. Dostopno na: http://php.net/

[23] (2014) Wikipedia o MySQL. Dostopno na: http://sl.wikipedia.org/wiki/MySQL

[24] (2014) Wikipedia about JSON. Dostopno na: http://en.wikipedia.org/wiki/JSON